RADAR TARGET GENERATION & DETECTION SENSOR FUSION PROJECT #4

Jasmine L. Taketa-Tran

OVERVIEW

The objective of this project was to simulate FMCW radar detection of a moving target, and then to perform signal processing functions to estimate the range and doppler velocity of the simulated target.

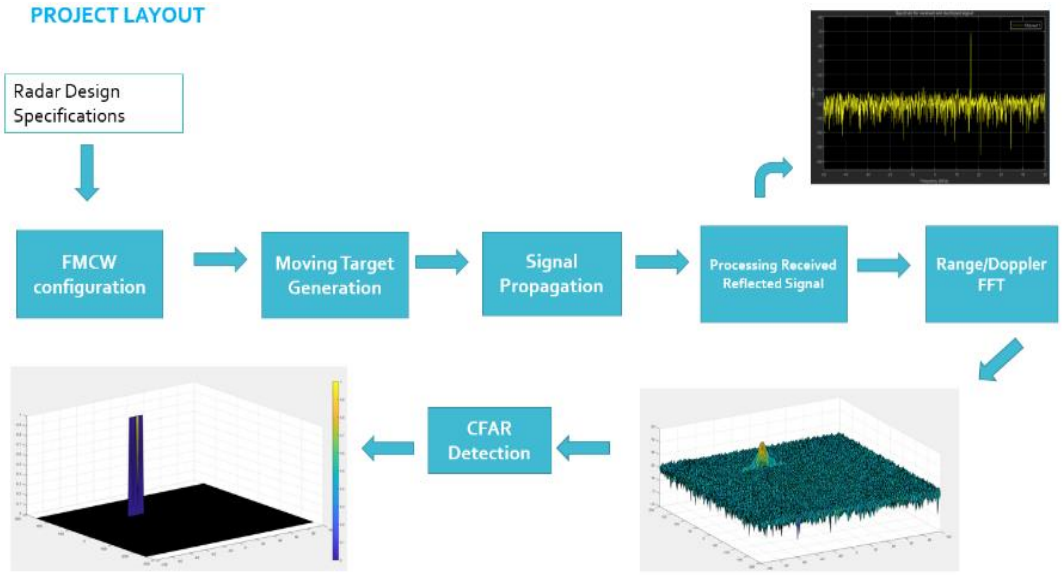

Figure 1: Project Workflow for Radar Simulation and Detection

FMCW WAVEFORM DESIGN

Given the systems requirements design specification, design a Frequency-Modulated Continuous-Wave (FMCW) radar waveform. Find the chirp bandwidth, time and slope.

Frequency	**77 GHz**
Range Resolution	1 m
Max Range	200 m
Max velocity	70 m/s
Velocity resolution	3 m/s

Figure 2: FMCW Radar Systems Requirements Design Specifications

```
%% Define Radar Specifications
% Configure the FMCW Waveform based on the System Requirements

fc= 77.0e9; % Operating Carrier Frequency (Hz)
c = 3.0e8; % Speed of Light (meters/sec)
lambda = c/fc; % Wavelength
Rmax = 200; % Maximum Detectable Range (meters)
Vmax = 70; % Maximum Velocity (meters/sec)
deltaR = 1.0; % Range Resolution in (meters)
deltaV = 3.0; % Velocity Resolution (meters/sec)
```

```
%% Compute FMCW Waveform Parameters
% Calculate the FMCW waveform parameters given the requirements specs

% Bandwidth of each chirp at the given range resolution:
Bsweep = c/(2*deltaR);

% Sweep Time for each chirp defined at 5.5x round trip time at max range:
Tsweep = 5.5*(Rmax*2/c);

% Slope of the chirp
Slope = Bsweep/Tsweep;

% The number of chirps in one sequence
Nd = 128; % #of doppler cells OR #of sent periods

% The number of samples on each chirp
Nr = 1024; % for length of time OR # of range cells
```

Based on radar specifications, the values of the chirp parameters are:

- Bandwidth: 1.5e8
- Time: 7.3333e-06
- Slope: 2.0455e+13

SIMULATION LOOP

Simulate target movement and calculate the beat or mixed signal for every timestamp. A beat signal should be generated such that once range FFT is implemented, it gives the correct range, i.e. the initial position of target assigned with an error margin of +/- 10 meters.

```
%% Model Signal Propagation for the Moving Target Simulation

% Set initial range and velocity of the target
targetRange = 110; %100; % Initial Distance to the Target (<= 200 m)
targetVelocity = 20; %-40; % Closing Velocity of Target (-70 to +70 m/s)

% Initialize vectors to store variable time histories
time = linspace(0,Nd*Tsweep,Nr*Nd); % time vector
Tx = zeros(1,length(time)); % transmit signal
Rx = zeros(1,length(time)); % receive signal
beatSig = zeros(1,length(time)); % beat frequency
range2Tgt = zeros(1,length(time)); % range to target
tau = zeros(1,length(time)); % time delay (trip time for the signal)
for tIdx = 1:length(time)

    % Displace the target based on the assumption of constant velocity
    range2Tgt(tIdx) = targetRange + (targetVelocity * time(tIdx));

    % Ensure that the targetRange does not exceed the maximum detectable
    % range of the radar. If it does, then flag it as undetectable.
    if range2Tgt(tIdx) > Rmax
        range2Tgt(tIdx) = NaN;
    end

    % Update the trip/delay time for the received signal
    tau(tIdx) = (range2Tgt(tIdx)*2)/c;

    % For each time stamp, update the transmit and receive signals
    % Receive signal is the time-delayed version of the transmit signal
    Tx(tIdx) = cos(2*pi*((fc*time(tIdx)) + ((Slope*(time(tIdx)^2))/2)));
    Rx(tIdx) = cos(2*pi*((fc*(time(tIdx)-tau(tIdx))) +...
        ((Slope*((time(tIdx)-tau(tIdx))^2))/2)));
end

% % Add 20% random noise to the signals
% Tx = Tx.*(1 + 0.2*randn(size(time)));
% Rx = Rx.*(1 + 0.2*randn(size(time)));

% For each displacement, determine the beat signal (frequency shift)
beatSig = Tx.*Rx;
```

For the moving target simulation, I experimented with 2 different range and relative velocity settings. In the first experiment, I used the initial target range of 110 meters and relative closing velocity of 20 meters-per-second. I also experimented with a target at range 100 meters and relative closing velocity of -40 meters-per-second. It was assumed for the purposes of this simple study that the velocity of the target

was constant for the duration of observation. Note that I did not add statistical noise to any of the signals. I was not sure how to model this for radar.

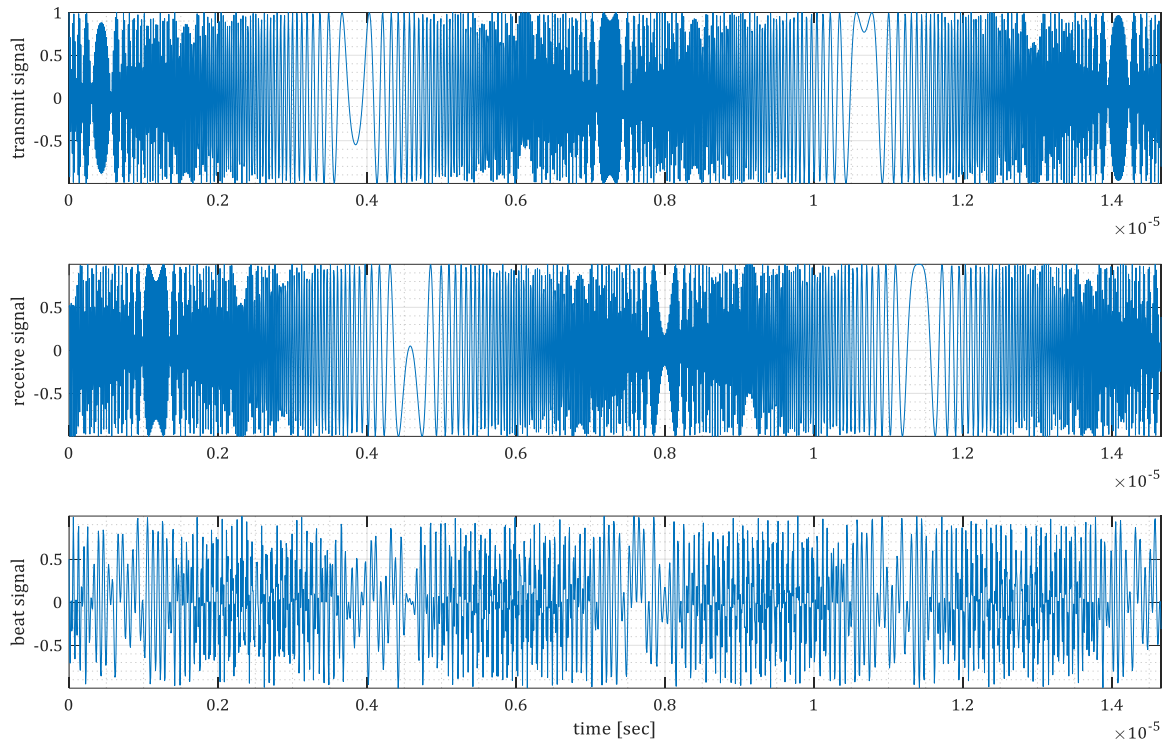

Figure 3: Two (out of 128) periods of a simulated target with initial range of 110m and closing velocity of 20m/s

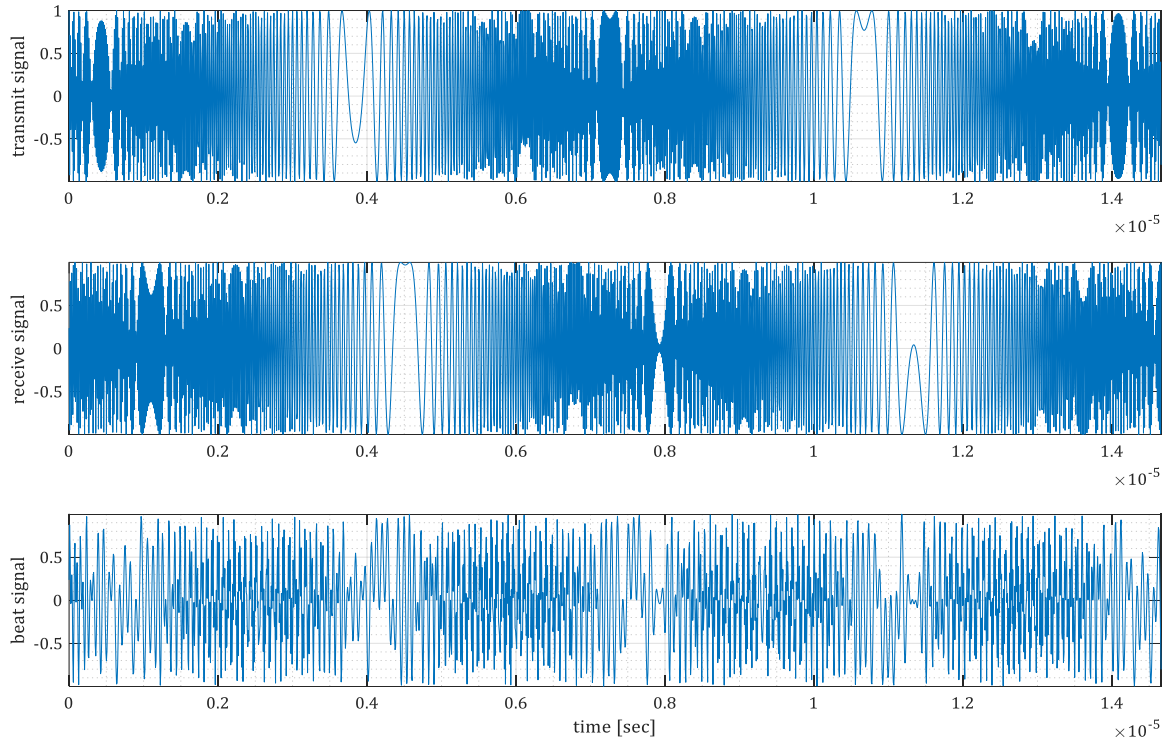

Figure 4: Two (out of 128) periods of a simulated target with initial range of 100m and closing velocity of -40m/s

RANGE FFT (1ST FFT)

Implement the Range FFT on the beat or mixed signal and plot the result. A correct implementation should generate a peak at the correct range, i.e. the initial position of the assigned target with an error margin of +/- 10 meters.

```
%% Estimate the Range to Target
% Perform 1D FFT on the beat signal to estimate the range to target

% Reshape the mix signal into number of range samples (Nr)
% and number of doppler samples (Nd) to form an Nr-by-Nd array
% Nr and Nd will also be used for the FFT sizes
beatMat = reshape(beatSig,[Nr Nd]);
range2TgtMat = reshape(range2Tgt,[Nr Nd]);

% Compute the normalized FFT of the beat signal along the range dimension
beat_fft = fft(beatMat,Nr)/Nr;

% Compute the amplitude of the normalized signal
beat_fft = abs(beat_fft);

% Compute the single-sided spectrum based on the even-valued signal length
beat_fft = beat_fft(1:Nr/2+1,:); % discard negative half of fft
```

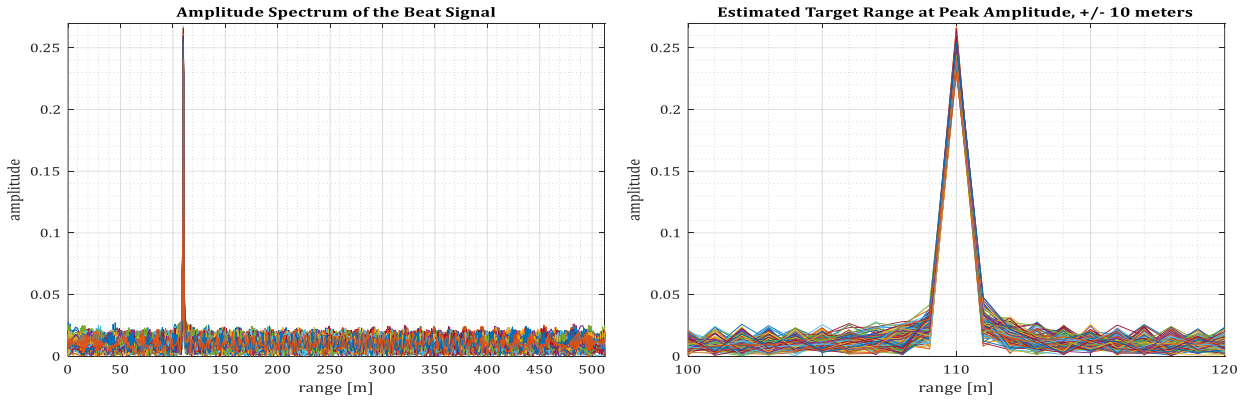

Figure 5: FFT for a Simulated Target at Initial Range of 110 m and Relative Velocity of 20 m/s

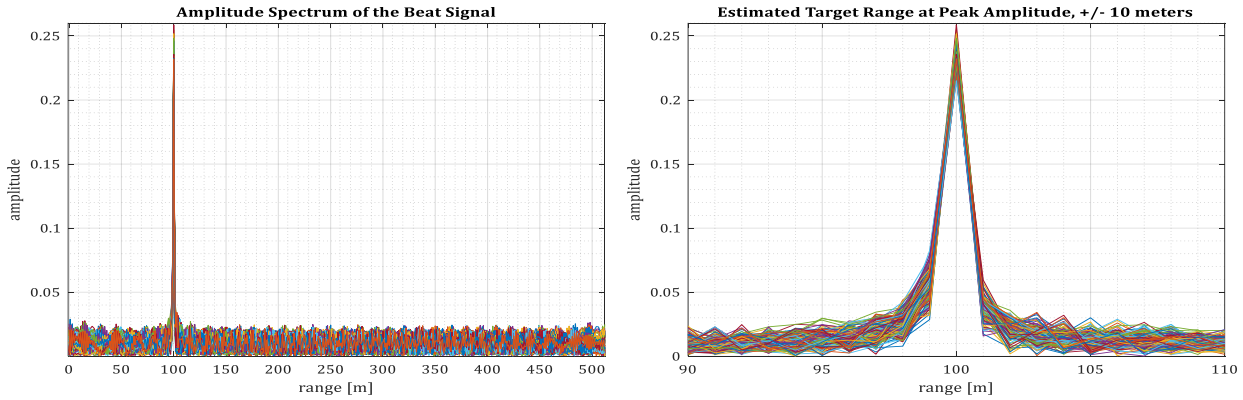

Figure 6: FFT for a Simulated Target at Initial Range of 100 m and Relative Velocity of -40 m/s

2D CONSTANT FALSE ALARM RATE

Implement the 2D CFAR process on the Range Doppler Map. The 2D CFAR processing should be able to suppress the noise and separate out the target signal.

```
%% Generate the Range Doppler Map (RDM)
% Perform a 2D FFT on the beat signal to generate a Range Doppler Map
% Perform CFAR processing on the output of the 2nd FFT to find the target
beatMat = reshape(beatSig,[Nr Nd]);

% 2D FFT using the FFT size for both dimensions
sig_fft2 = fft2(beatMat,Nr,Nd);

% Take just one side of signal in the range dimension
sig_fft2 = sig_fft2(1:Nr/2,1:Nd);
sig_fft2 = fftshift(sig_fft2);
RDM = abs(sig_fft2);
RDM = 10*log10(RDM);
```

```
%% Perform CFAR Thresholding on the RDM to Detect the Target
% Note: For a Target at 100m, Gr = 6, Gd = 22 & tnr = 9
%       For a Target at 110m, Gr = 5, Gd = 23 & tnr = 10

% Select the Number of Training Cells
Tr = 20; % Number of Training Cells in the Range Dimension
Td = 10; % Number of Training Cells in the Doppler Dimension

% Select the Number of Guard Cells
Gr = 5;  % Number of Guard Cells in the Range Dimension
Gd = 23; % Number of Guard Cells in the Doppler Dimension

% Offset the threshold by SNR value in dB
tnr = 10; % Define the Threshold-to-Noise Ratio

% Slide the window filter across the Range Doppler Map
% Allow an outer image buffer for Guard & Training cells
for i = Tr+Gr+1:(Nr/2)-(Gr+Tr)
    for j = Td+Gd+1:Nd-(Gd+Td)
```

```
        % Step through the bins & grid surrounding the Cell Under Test
        noise_level = zeros(1,1);
        for p = i-(Tr+Gr):i+Tr+Gr
            for q = j-(Td+Gd):j+Td+Gd

                % Exclude the Guard Cells and the Cell Under Test
                if (abs(i-p)>Gr) || (abs(j-q)>Gd)

                    % Convert the logarithmic dB value to linear power
                    % Then sum the noise across the Training Cells
                    noise_level = noise_level + db2pow(RDM(p,q));
                end
            end
        end

        % Compute threshold based on the noise and offset (tnr)
        numGridCells = ((2*Tr)+(2*Gr)+1)*((2*Td)+(2*Gd)+1);
        numGuardCells = Gr*Gd;
        numTrainCells = numGridCells - numGuardCells - 1;

        % Estimate the average noise across the Training Cells
        % Convert back to logarithmic dB after obtaining the average
        threshold = pow2db(noise_level/numTrainCells);
        threshold = threshold + tnr; % multiplying in linear space

        % Compare the Cell Under Test to the Detection Threshold
        CUT = RDM(i,j); % get signal within the Cell Under Test
        if CUT < threshold % if it is below threshold
            RDM(i,j) = 0; % then assign it a value of 0
%       else % if it is above threshold
%           RDM(i,j) = 1; % then assign it a value of 1
        end
    end
end

% Suppress edges of the Range Doppler Map to account for the presence
% of Training and Guard Cells surrounding the Cell Under Test
RDM_thresholded = zeros(size(RDM));
RDM_thresholded(Tr+Gr+1:(Nr/2)-(Gr+Tr),Td+Gd+1:Nd-(Gd+Td)) = ...
    RDM(Tr+Gr+1:(Nr/2)-(Gr+Tr),Td+Gd+1:Nd-(Gd+Td));

% Plot the filtered RDM as a function of range and velocity
figure; surf(doppler_axis,range_axis,RDM_thresholded);
title('Range Doppler Map after CFAR Thresholding');
xlabel('doppler'); ylabel('range'); axis tight;
set(gca,'FontName','Cambria'); grid on; grid minor;
```

For a simulated target at Initial Range of 110 meters and Relative Velocity of 20 meters/second:

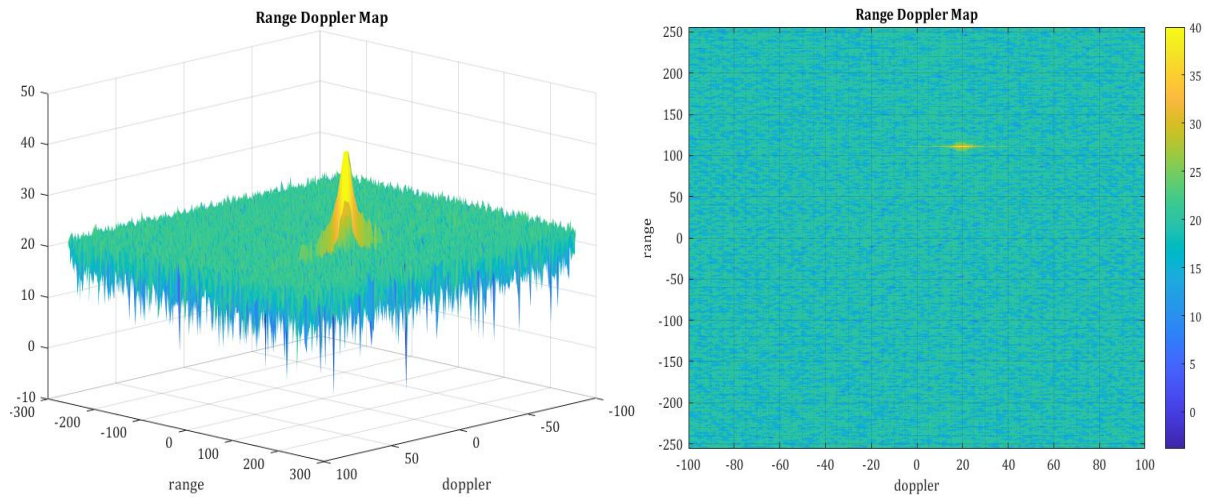

Figure 7: Range Doppler Map before thresholding

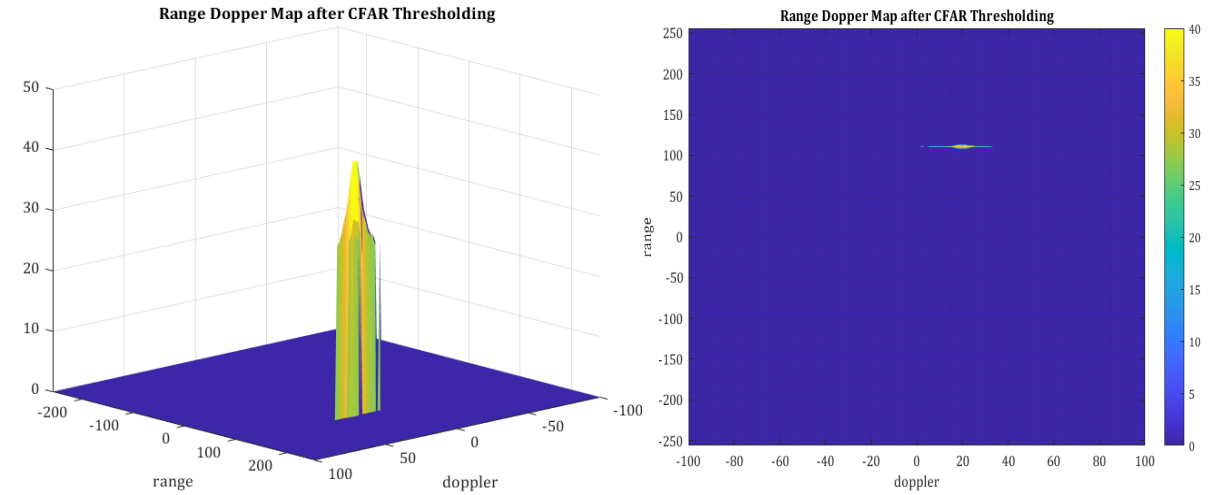

Figure 8: Range Doppler Map after CFAR-based thresholding

Here is a closer look at the target for the same images on the previous page:

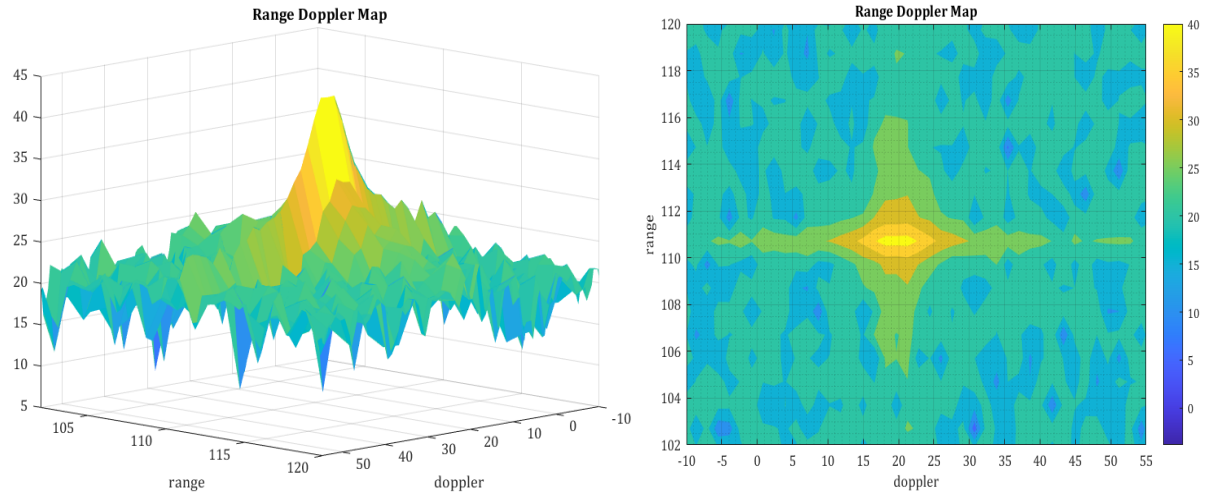

Figure 9: Zoomed-in image of target on Range Doppler Map before thresholding

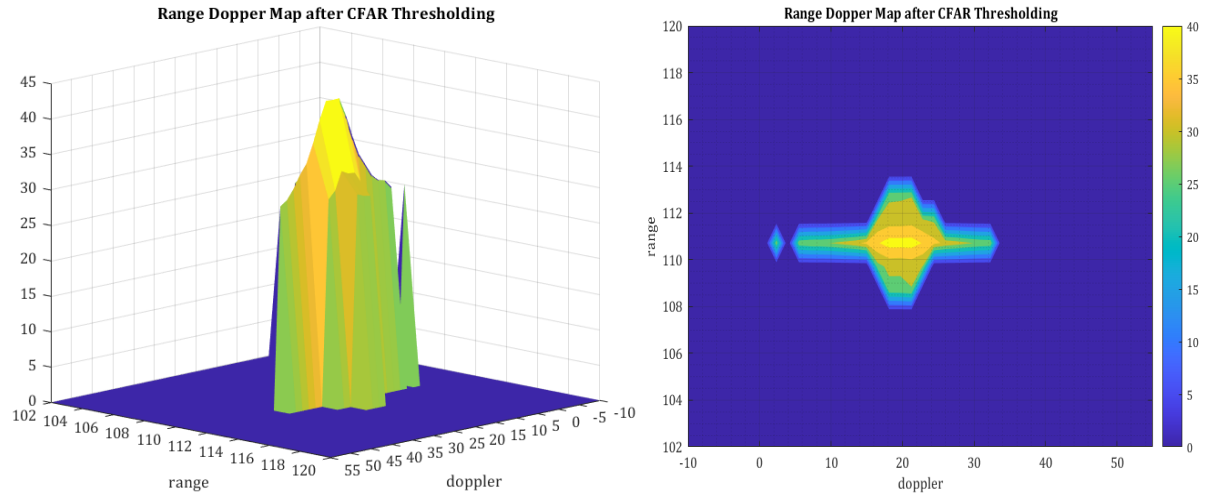

Figure 10: Zoomed-in image of target on Range Doppler Map after thresholding

For a simulated target at Initial Range of 100 meters and Relative Velocity of -40 meters/second:

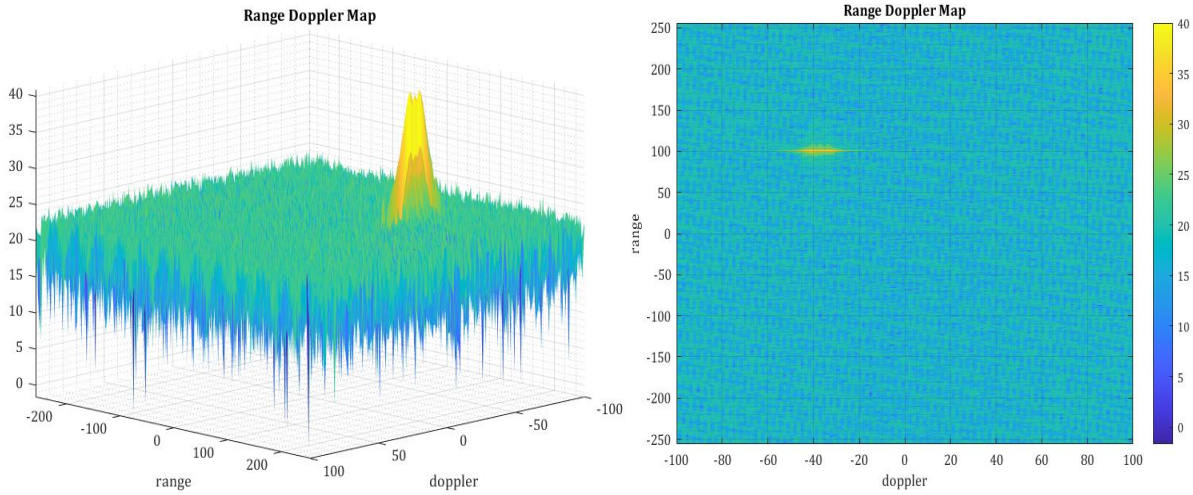

Figure 11: Range Doppler Map before thresholding

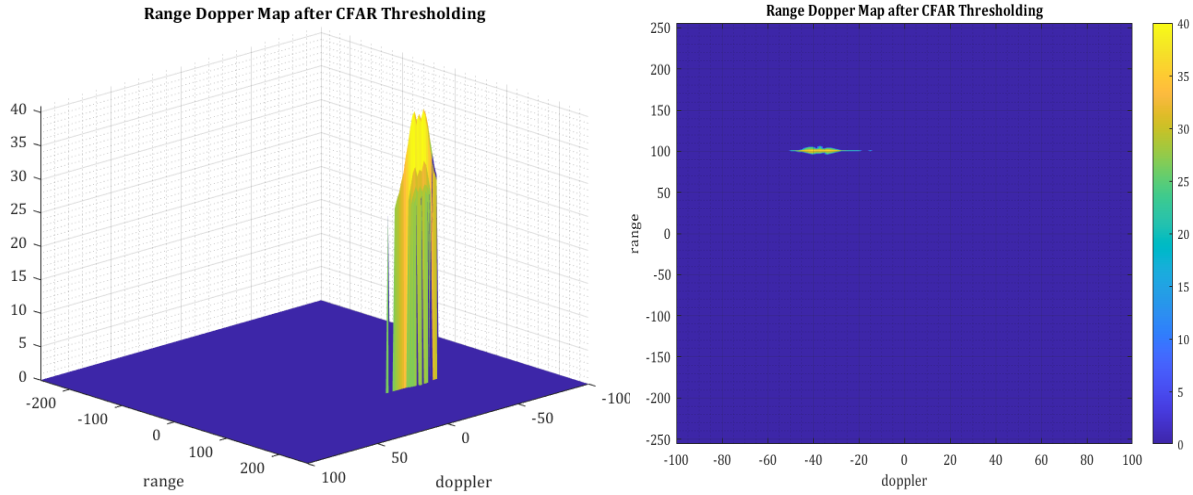

Figure 12: Range Doppler Map after CFAR-based thresholding

Here is a closer look at the target for the same images on the previous page:

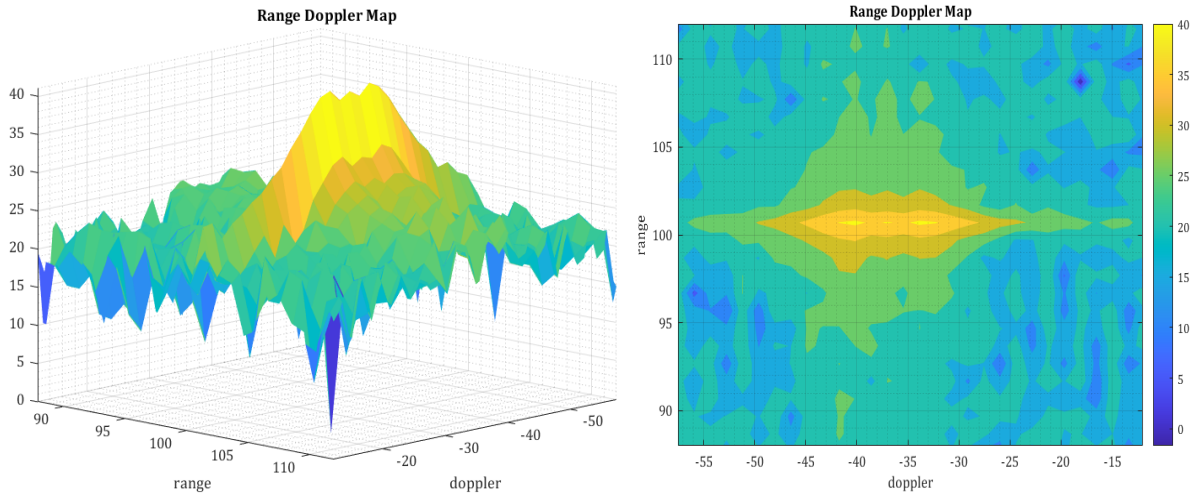

Figure 13: Zoomed-in image of target on Range Doppler Map before thresholding

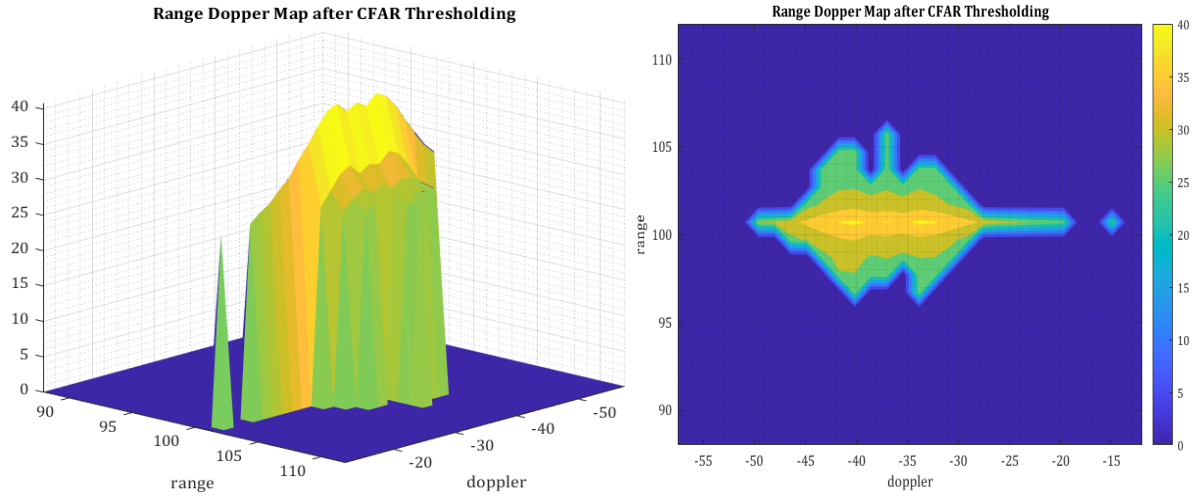

Figure 14: Zoomed-in image of target on Range Doppler Map after thresholding

CREATE A CFAR README FILE

Write brief explanations for the following:

- *Implementation steps for the 2D CFAR process*
- *Selection of Training Cells, Guard Cells and Offset*
- *Steps taken to suppress the non-thresholded cells at the edges*

Two-dimensional CFAR processing is performed on the Range Doppler Map, which is generated by taking the FFT of the beat signal in each chirp to determine the target range over time, followed by a second FFT on the range FFT over time to determine the rate of change of phase across the chirps. Hence, the Range Doppler Map represents our estimate of the range of the target vs the Doppler velocity over time.

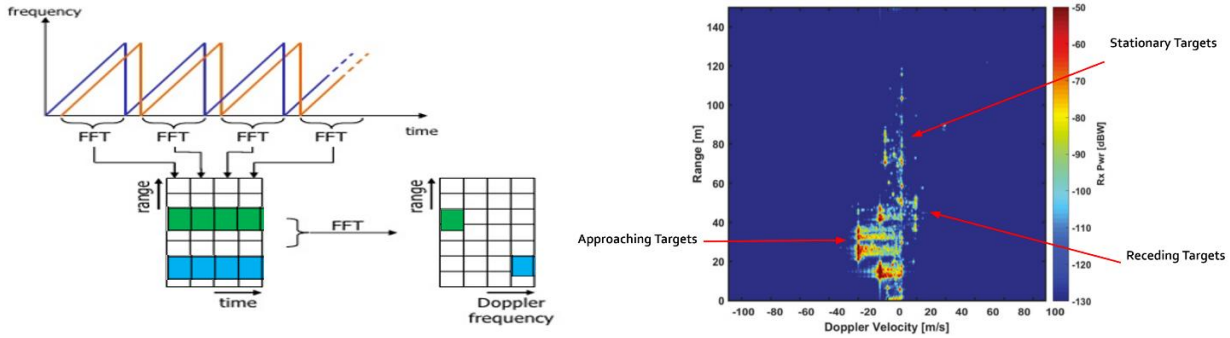

Figure 15: Generating the Range Doppler Map

The 2D CFAR process is a dynamic thresholding method used to detect target(s) of interest in the presence of unwanted clutter. The threshold is computed on a bin-by-bin basis by estimating the noise level in surrounding bins and multiplying an offset to it. The idea is to adaptively vary the threshold based on estimated noise levels to maintain a constant false alarm rate that is independent of the clutter and includes the true target. The threshold is compared to the range-doppler value of the bin to determine whether a target is present. This process is repeated for all bins in the Range Doppler Map (minus an outer border) using a sliding window approach.

The sliding window consisted of 3 parts. We called each bin as it was being evaluated the Cell Under Test (CUT). Surrounding the CUT was a region we called the Guard Cells, which were used to allow for target energy crossing multiple bins in the Range Doppler Map. This was necessary to avoid corrupting the local noise level estimate in the next step. We then generated an estimate of the local noise level by computing the sample mean of the bins in a local annulus region surrounding the CUT and Guard Cells – these were referred to as the Training Cells. One small detail: Training Cell values needed to be converted from logarithmic dB to linear power before computing the sample mean, then converted back to dB for comparing the threshold against the range-Doppler value in the CUT. The best values for these parameters were application/problem specific. For this project, I manually tuned the number of Guard Cells and number of Training Cells on a case-by-case basis by looking at the two-dimensional target extent in the Range Doppler Map:

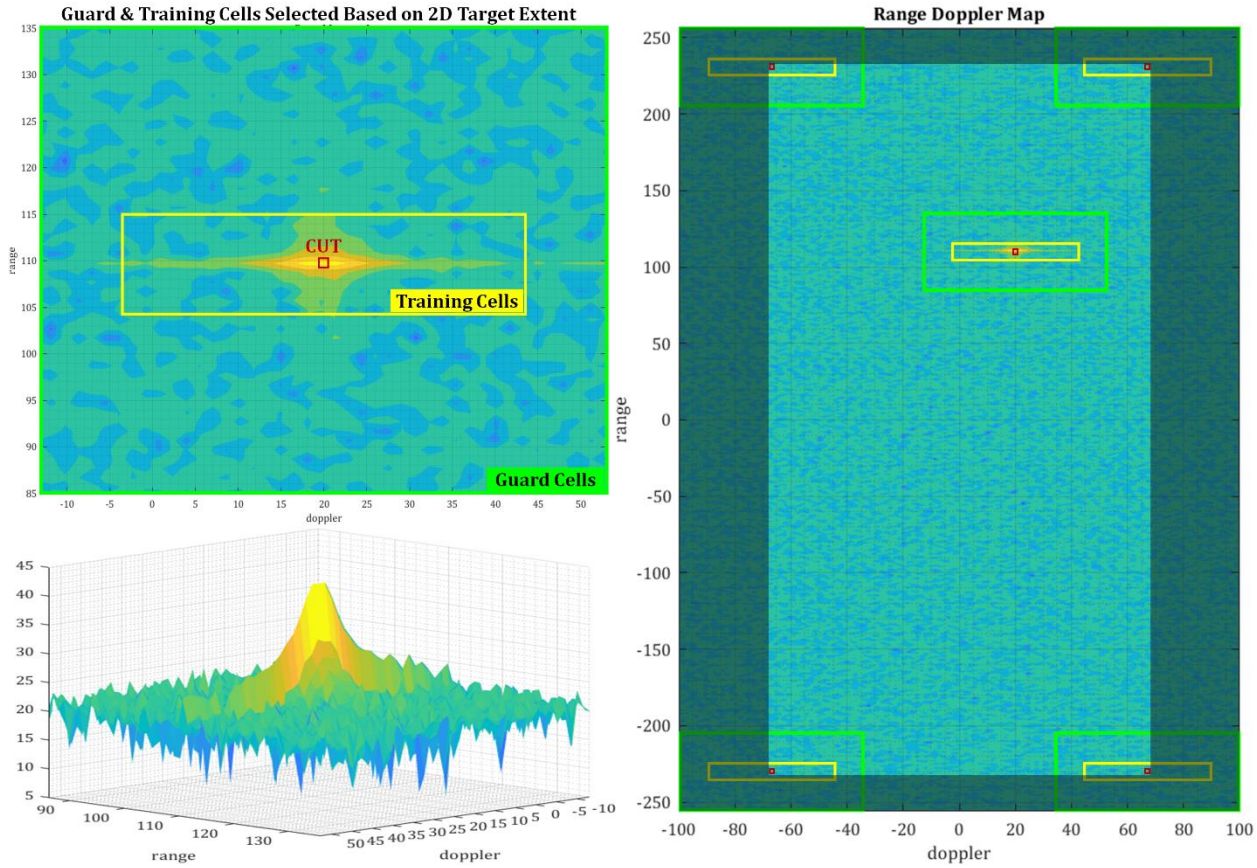

Figure 16: For a target at initial range of 110 m with closing velocity of 20 m/s, the following window parameters were selected: 5 guard cells in the range dimension (Gr), 23 guard cells in the Doppler dimension (Gd), 20 training cells in the range dimension (Tr), and 10 training cells in the Doppler dimension (Td).

Next, an offset term that I referred to as the “threshold-to-noise ratio” (tnr) was added to the estimated noise level in log dB space to produce the detection threshold. We then compared the signal in the CUT against this threshold. If the signal in the CUT fell below this threshold, we suppressed it in the Range Doppler Map by setting it to zero. If the signal in the CUT rose above the threshold, I kept the value as-is (note: the project steps suggested setting it to 1, but I preferred to see the target distribution instead).

For selection of the offset, I used a trial-and-error process that consisted of visualizing the CFAR detection results for multiple offset values, then picking the one that maximized true target detection while minimizing the number of false detections. For the 110 m target with 20 m/s closing velocity and corresponding guard and training cell parameters, I selected an offset of 10. CFAR detection results across a parametric sweep of offsets for the 110 m target are shown below:

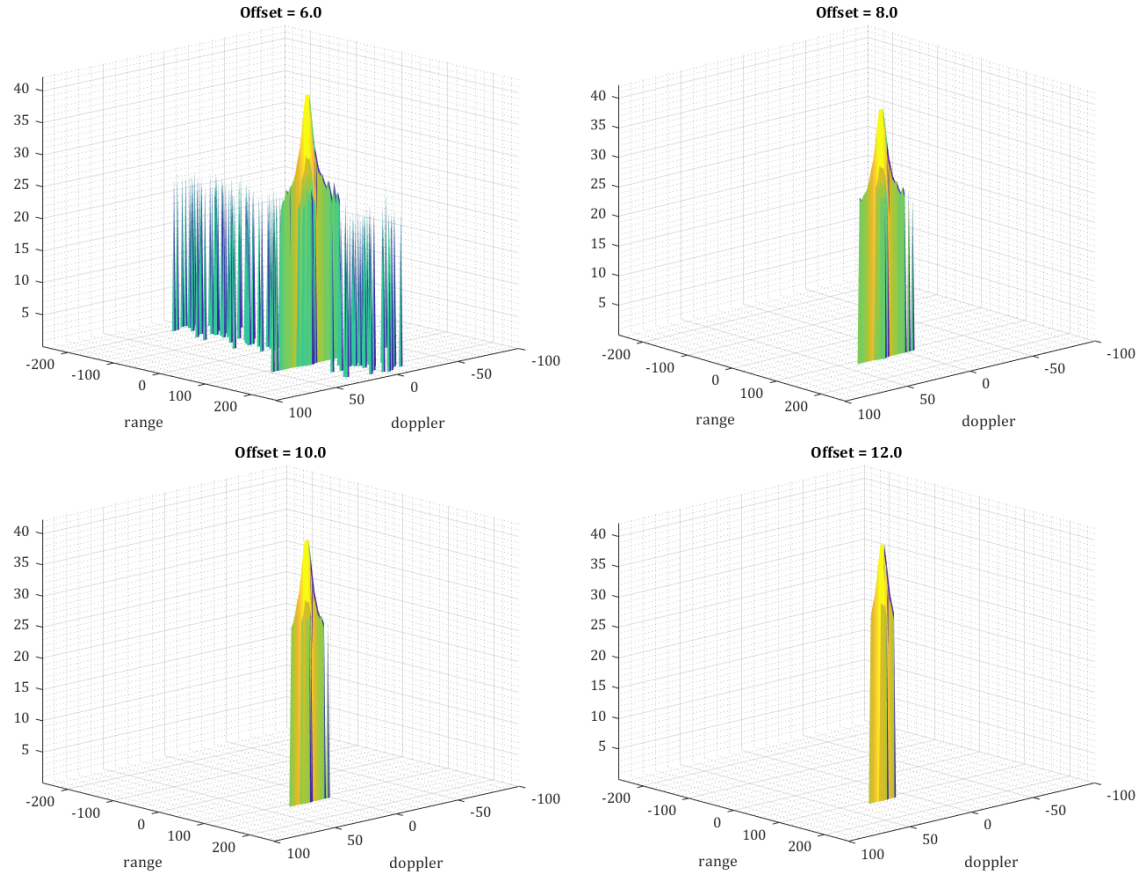

Figure 17: CFAR detection results based on multiple offsets for the 110 m target with -20 m/s closing velocity

For the 100 m target with -40 m/s closing velocity, I selected 6 guard cells in the range dimension, 22 guard cells in the Doppler dimension, 20 training cells in the range dimension, 10 training cells in the Doppler dimension, and a threshold offset of 9.0. This produced the results seen in the previous section.

Another tradeoff that I had to consider was the inverse relationship between the size of the window and the effective field-of-view imposed by this window in the Range Doppler Map. Targets close to the edge could potentially be missed if the selected window size is too large. On the other hand, too few training cells could produce sub-optimal estimates of the local noise level, and too few guard cells could enable leakage of target signal into the training cells, corrupting the noise estimate. Hence, window size should be taken into consideration when evaluating the overall sensor coverage (detection range vs field-of-view) of a vehicle.

Cells in the outer border of the Range Doppler Map had to be suppressed according to the size of the sliding window. We accounted for this by skipping Gr+Tr bins along the max and min range dimension, and Gd+Td bins along the max and min Doppler dimension, when looping over the Cells Under Test. The reduced-sized output of the CFAR processing was then mapped back to its original position in a zero array that was equivalent in size to the full Range Doppler Map:

```
% Suppress edges of the Range Doppler Map to account for the presence
% of Training and Guard Cells surrounding the Cell Under Test
RDM_thresholded = zeros(size(RDM));
RDM_thresholded(Tr+Gr+1:(Nr/2)-(Gr+Tr),Td+Gd+1:Nd-(Gd+Td)) = ...
    RDM(Tr+Gr+1:(Nr/2)-(Gr+Tr),Td+Gd+1:Nd-(Gd+Td));
```

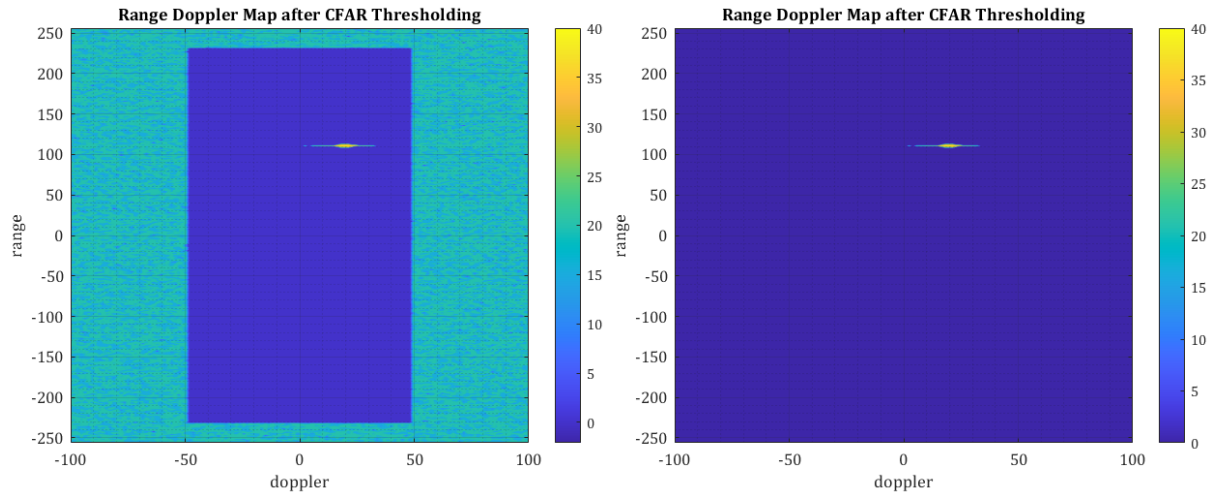

Figure 18: Suppressing the non-thresholded cells on the border of the Range Doppler Map